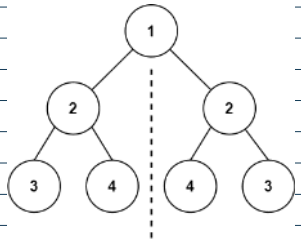
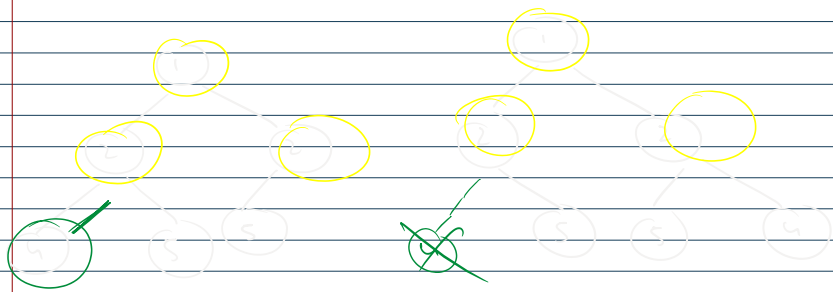
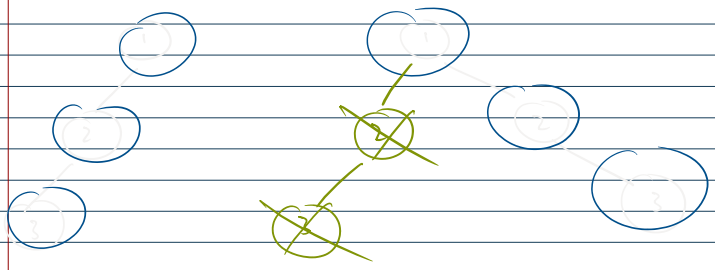
Symmetric Tree





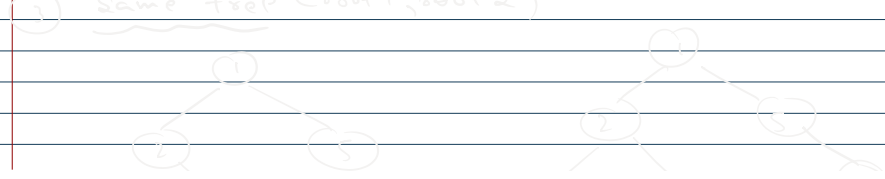


Symmetric by values + Structure

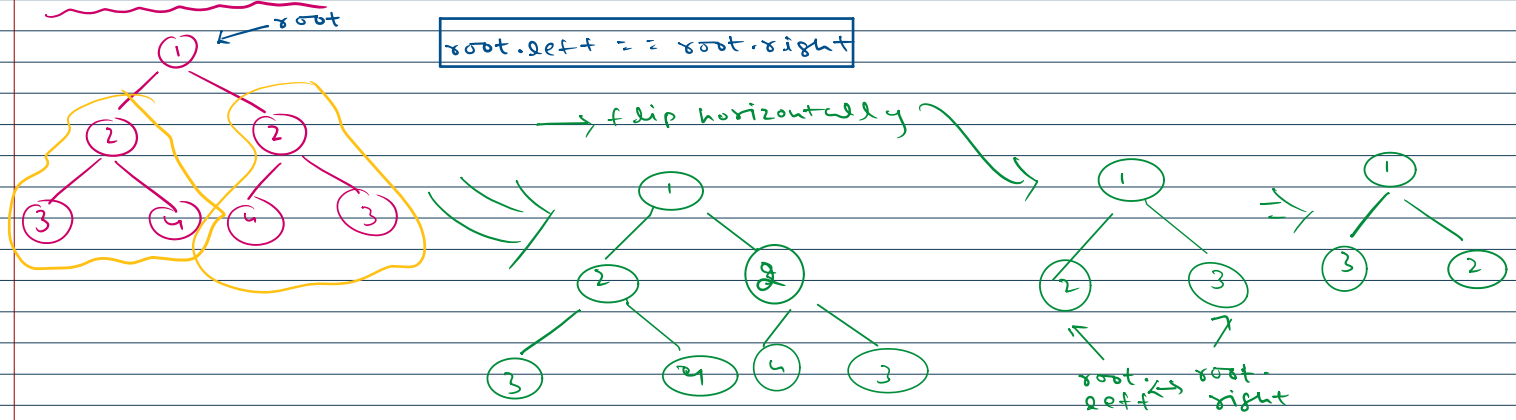
① Symmetric Tree (root)

② Mirror Tree (root1, root2)

③ Same tree (root1, root2)

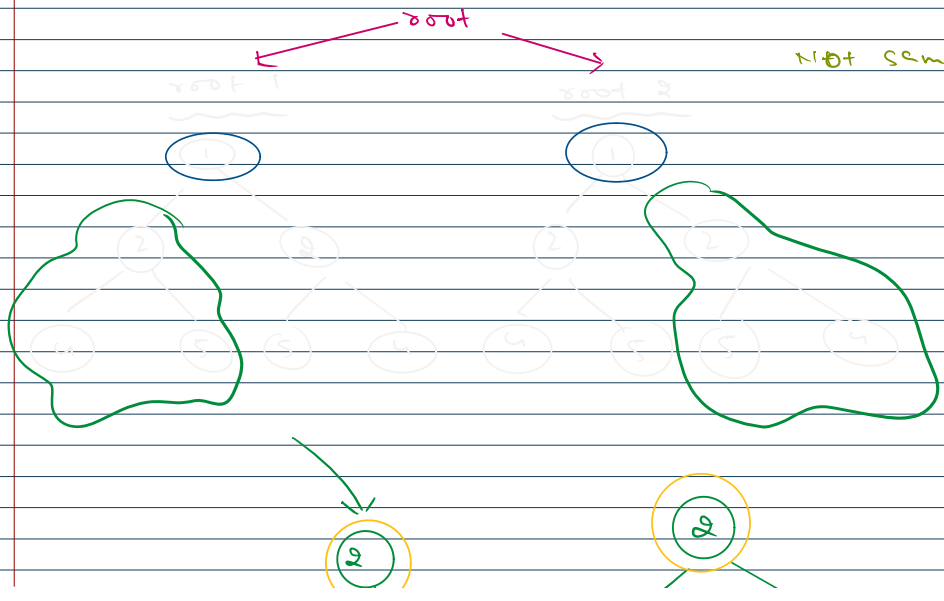


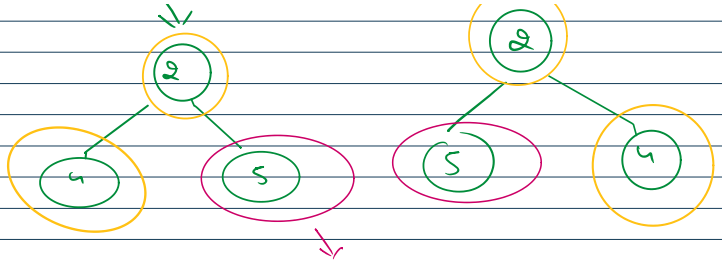
Symmetric Tree



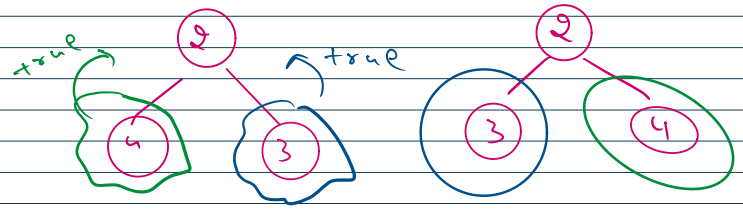
root 1, root 2 ⇒ same tree

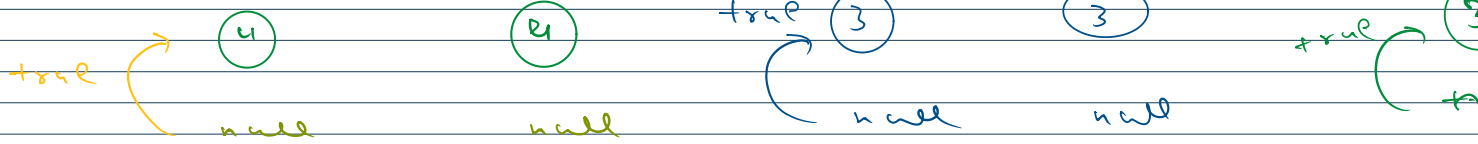
root 1	root 2	
null	null	⇒ Same tree (true)
1	null	⇒ Not same tree (false)
null	2	⇒ Not same tree (false)
1	1	⇒ Same ⇒ if root1.val == root2.val ⇒ true ⇒ Not same ⇒ else ⇒ false



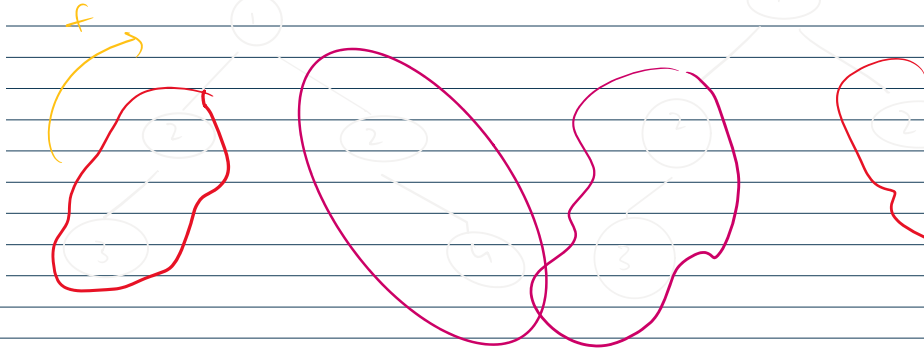


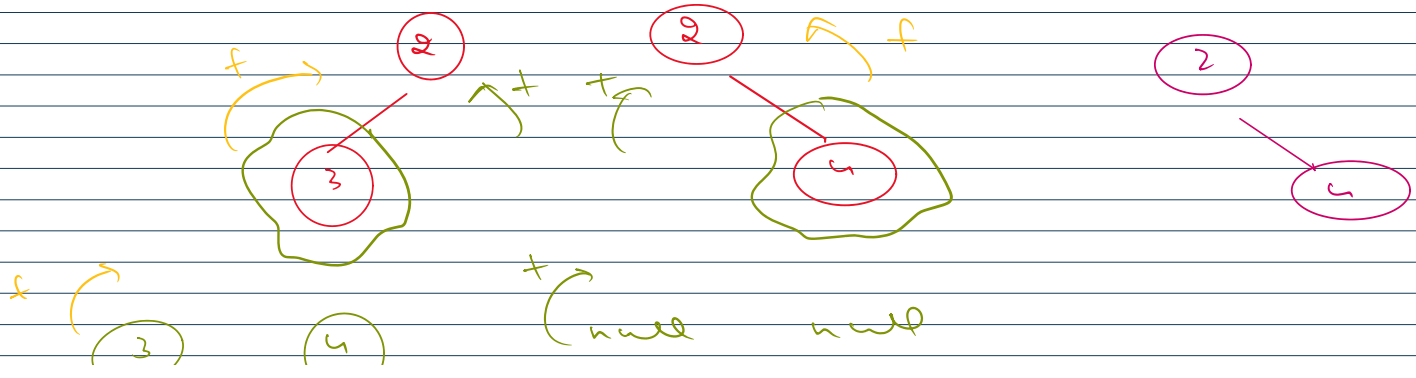
```java
public boolean solve(TreeNode root1, TreeNode root2) {
    if(root1 == null && root2 == null) { return true; }
    if(root1 == null && root2 != null) { return false; }
    if(root1 != null && root2 == null) { return false; }
    if(root1.val != root2.val) { return false; }

    boolean left = solve(root1.left, root2.right);
    boolean right = solve(root1.right, root2.left);

    return left && right;
}
```

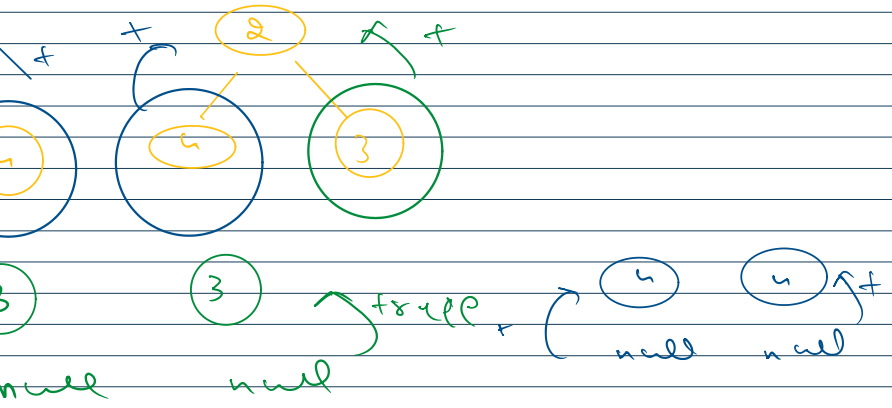


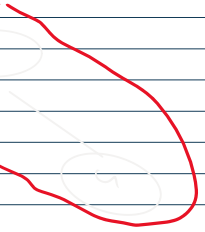


```java
public boolean solve(TreeNode root1, TreeNode root2) {
    if(root1 == null && root2 == null) { return true; }
    if(root1 == null && root2 != null) { return false; }
    if(root1 != null && root2 == null) { return false; }
    if(root1.val != root2.val) { return false; }

    boolean left = solve(root1.left, root2.right);
    boolean right = solve(root1.right, root2.left);

    return left && right;
}
```

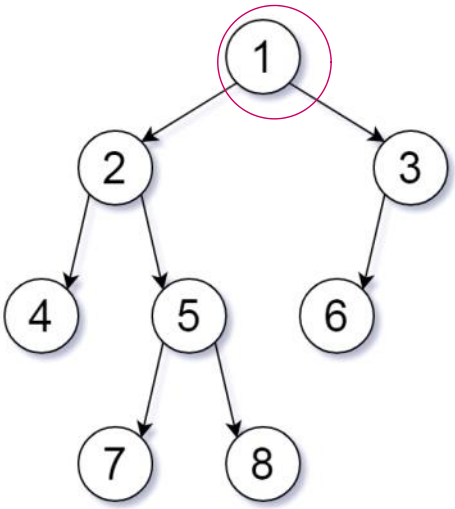


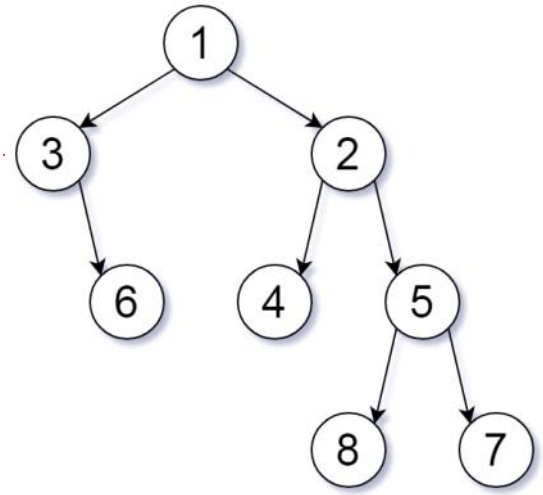


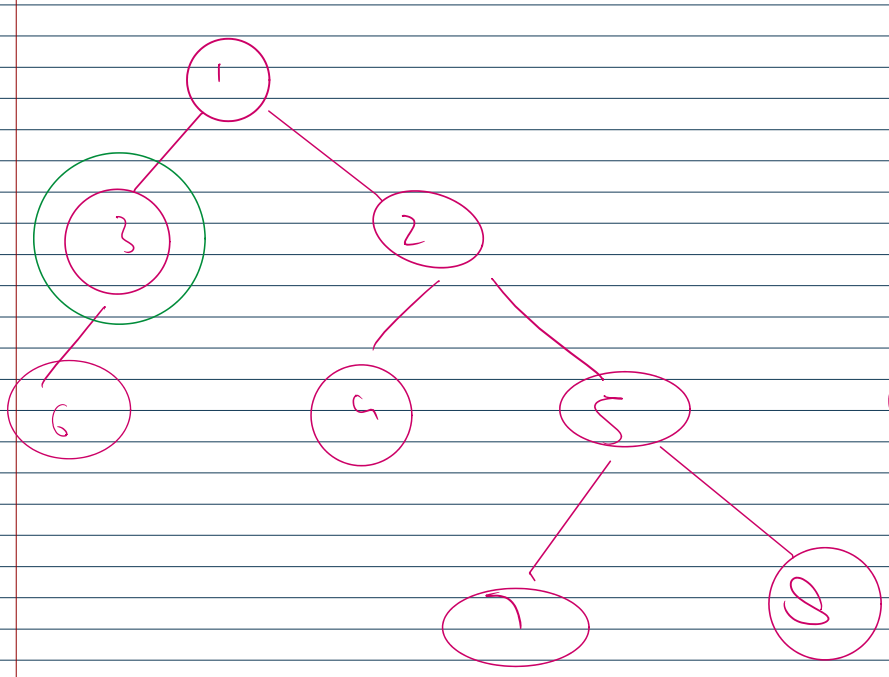

2
4
3
3
4
4
null
null
null

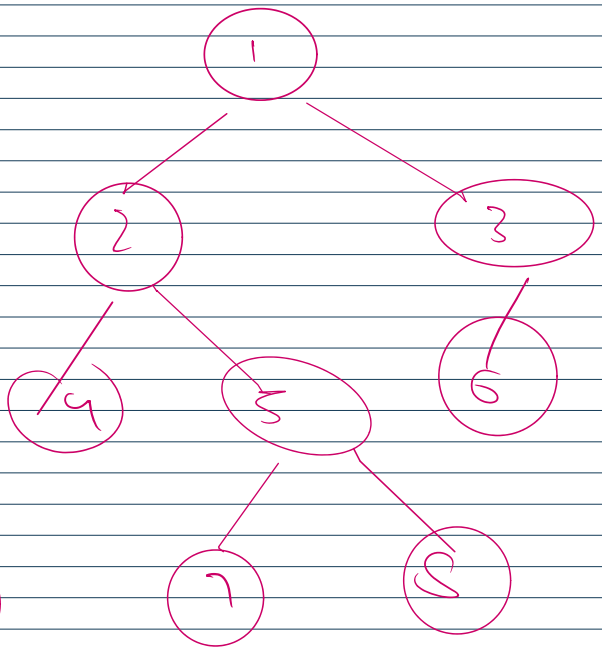


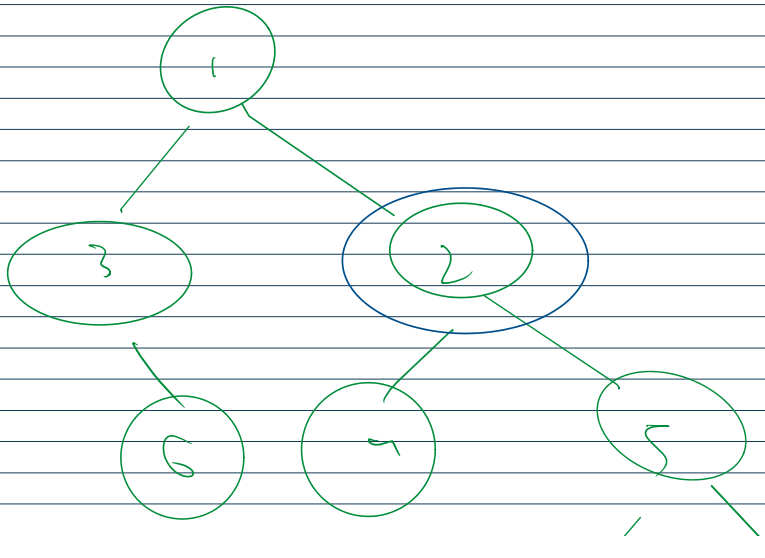

2
3

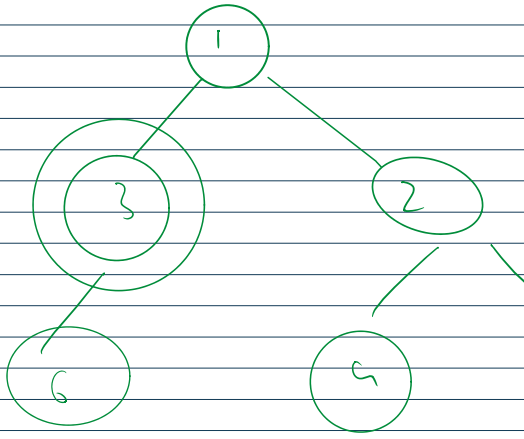

1
2
3
4
5
6
7
8

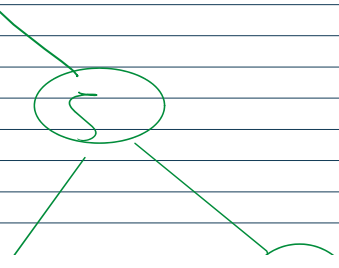

1
3
2
6
4
5
8
7

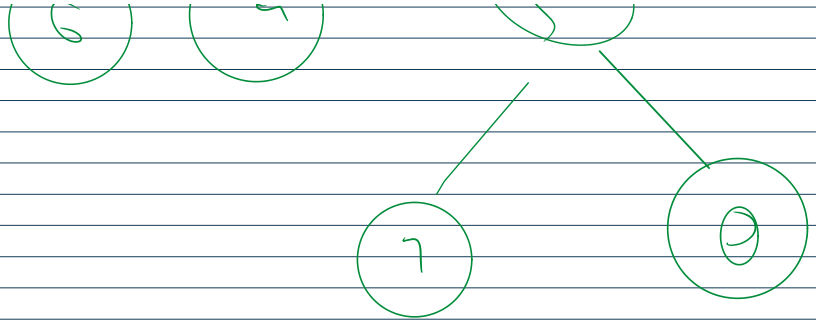


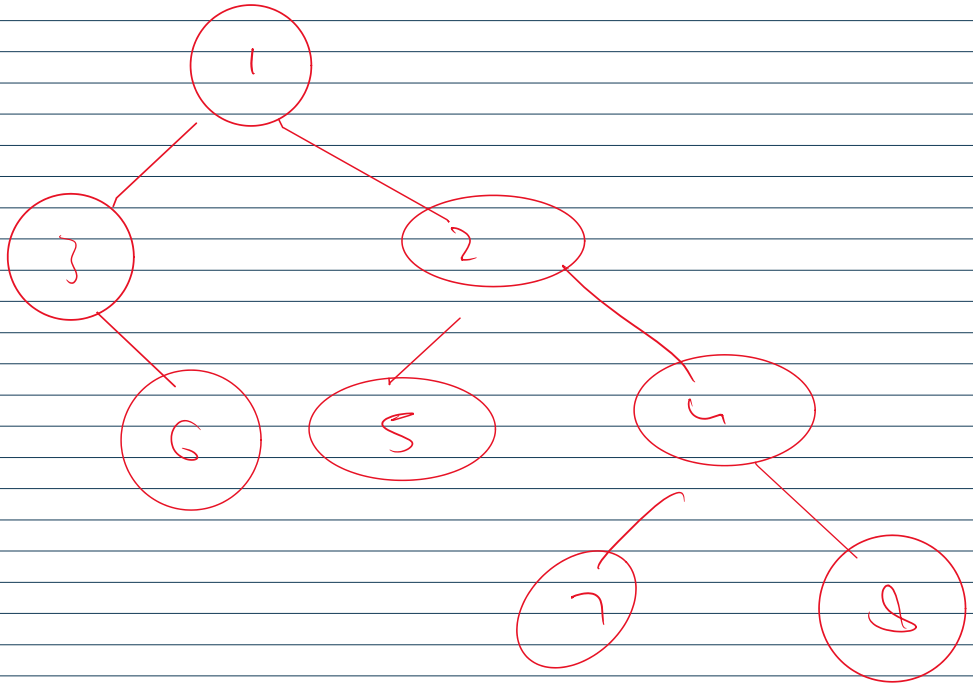




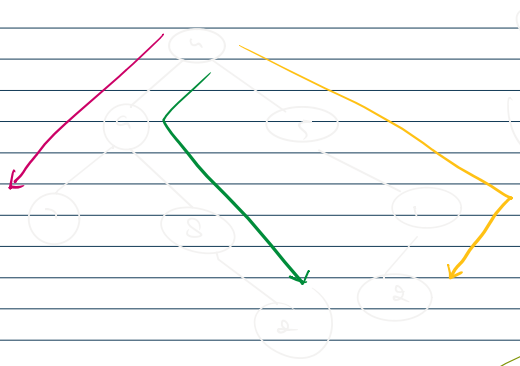




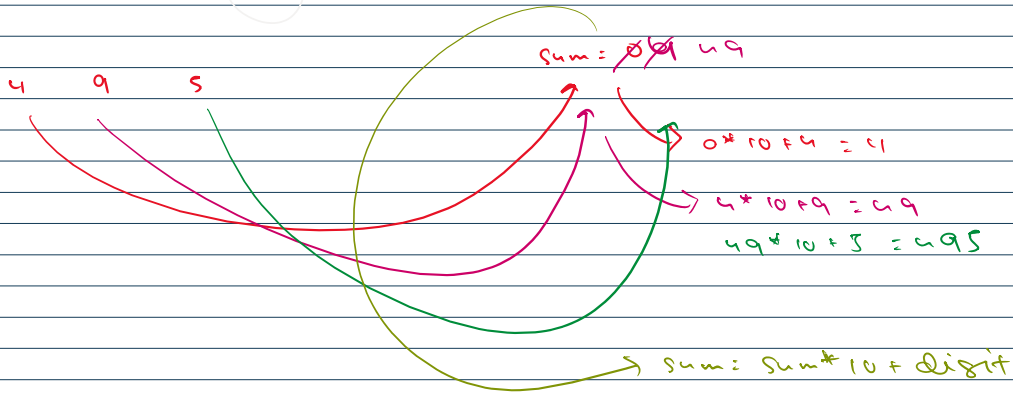


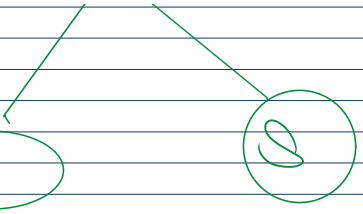


Sum root to leaf numbers



497
498
512



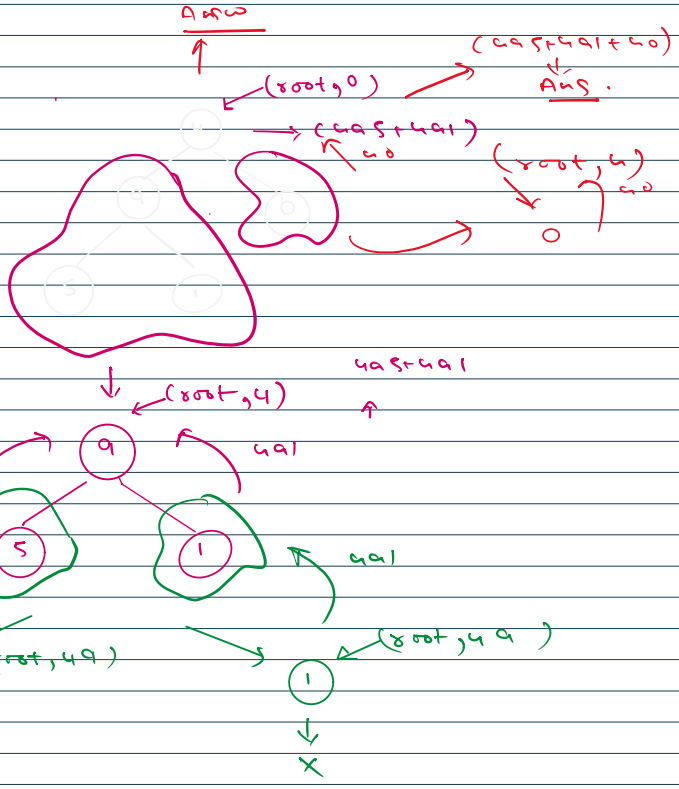


```
class Solution {
    public int sumNumbers(TreeNode root) {
        List<Integer> ans = new ArrayList<>();
        int sum = 0;
        solve(root, sum, ans);
        int total = 0;
        for(int i=0; i<ans.size(); i++) {
            total = total + ans.get(i);
        }
        System.out.println(ans);
        return total;
    }
    public void solve(TreeNode root, int sum, List<Integer> ans) {
        if(root == null) {
            return;
        }
        sum = sum * 10 + root.val;
        if(root.left == null && root.right == null) {
            ans.add(sum);
        }
        solve(root.left, sum, ans);
        solve(root.right, sum, ans);
    }
}
```

```
class Solution {
    public int sumNumbers(TreeNode root) {
        String op = "";
        List<String> ans = new ArrayList<>();
        solve(root, op, ans);
        int sum = 0;
        for(int i=0; i<ans.size(); i++) {
            sum = sum + Integer.parseInt(ans.get(i));
        }
        System.out.println(ans);
        return sum;
    }
    public void solve(TreeNode root, String op, List<String> ans) {
        if(root == null) {
            // ans.add(op);
            return;
        }
        op = op + root.val;
        if(root.left == null && root.right == null) {
            ans.add(op);
        }
        solve(root.left, op, ans);
        solve(root.right, op, ans);
    }
}
```

```
class Solution {
    public int sumNumbers(TreeNode root) {
        int sum = 0;
        return solve(root, sum);
    }
    public int solve(TreeNode root, int sum) {
        if(root == null) {
            return 0;
        }
        sum = sum * 10 + root.val;
        if(root.left == null && root.right == null) {
            return sum;
        }
        int sum1 = solve(root.left, sum);
        int sum2 = solve(root.right, sum);
        return sum1 + sum2;
    }
}
```

```
public int solve(TreeNode root, int sum) {
    if(root == null) {
        return 0;
    }
    sum = sum * 10 + root.val;
    if(root.left == null && root.right == null) {
        return sum;
    }
    int sum1 = solve(root.left, sum);
    int sum2 = solve(root.right, sum);
    return sum1 + sum2;
}
```




```
class Solution {
    public boolean hasPathSum(TreeNode root, int targetSum) {
        int sum = 0;
        return solve(root, sum, targetSum);
    }
    public boolean solve(TreeNode root, int sum, int targetSum) {
        if(root == null) {
            return false;
        }
        sum = sum + root.val;
        if(root.left == null && root.right == null) {
            return sum == targetSum;
        }
        return solve(root.left, sum, targetSum) || solve(root.right, sum, targetSum);
    }
}
```

```
class Solution {
    public boolean hasPathSum(TreeNode root, int targetSum) {
        return solve(root, targetSum);
    }
    public boolean solve(TreeNode root, int targetSum) {
        if(root == null) {
            return false;
        }
        targetSum = targetSum - root.val;
        if(root.left == null && root.right == null) {
            return targetSum == 0;
        }
        return solve(root.left, targetSum) || solve(root.right, targetSum);
    }
}
```